REMARKS

This amendment is being submitted to expedite the patent application process. Applicants have canceled all previously-pending claims and added new claims. In particular, by virtue of this amendment, claims 21-25 are pending. Claims 1-20 have been canceled without prejudice or disclaimer. New claims 21-25 have been added. It is submitted that the application, as amended, is in condition for allowance. Allowance of the pending claims are respectfully requested.

Claims 1-18 were rejected under 35 U.S.C. § 101 for being directed to non-statutory subject matter. Claims 1-18 have been canceled so this rejection is moot. Further, Applicant respectfully submits that the new claims produce a useful, tangible, and concrete result. For example, new claim 21 recites "executing, via the control module, the source code received from the remote computer; and reading and modifying, via the control module in response to the executing step, state information of the web browser as instructed by the source code." Additionally, new claim 22 recites a "computer readable tangible storage medium encoded with a program for providing support to a user from a remote computer, the program comprising instructions for execution by a processing circuit." Additionally, new claim 23 recites a "control module for embedding in a web page to allow support to be provided to a user from a remote computer, and new claim 25 recites a "customer service computer for providing support to a user of a remote computer."

Claims 1-3, 5-8, 10-14, 16, and 17 were rejected under 35 U.S.C. § 102(b) as being anticipated by Herrmann (U.S. Patent No. 5,995,756). Claims 4, 9, 15, and 18 were rejected under 35 U.S.C. § 103(a) as being unpatentable over Herrmann in view of Lee (U.S. Patent Application Publication No. 2003/0146937). Claims 1-18 have been canceled so this rejection is moot.

Claims 21-25 have been added by this amendment, and are provided to further define the invention disclosed in the specification. Claims 21-25 are allowable over Herrmann and Lee for at least the following reasons.

The present invention is directed to an efficient and easy-to-implement method and system for providing customer support to users interacting with a web page. One embodiment of the present invention provides a method for providing support to a user from a remote computer. According to this method, a control module embedded in a web page that is currently being viewed by a web browser is activated, and in response source code is received from the remote computer. The control module is embedded in the webpage by the web browser when the webpage is loaded into the browser, and the source code includes instructions directing the control module to perform at least one operation on the web browser. The control module is used to execute the source code received from the remote computer, and state information of the web browser is read and/or modified via the control module as instructed by the source code. In one exemplary embodiment, the state information of the web browser includes form information entered into at least one form field of the web page that is currently being viewed by the web browser.

This method for providing support takes away the need to build a custom client customer service application into the web interface. Instead, a generic control can be embedded into a web page, and this control can interact with a server side customer support application. This allows the customer service interface to be designed specifically for assisting users, and it can even be designed after the end user interface has already been deployed. Additionally, multiple customer service interfaces can exist simultaneously, each for assisting users with a specific aspect of a web application. Further, there is no need to change web pages or web sites to change the customer support application. A generic control can be embedded into a web page and a server side customer support application can be separate from the embedded control so that the update of one of these components does not necessarily require an update of the other. This method also provides additional security.

The Herrmann reference is directed toward a development system that provides a form-based development environment for partitioning an application. This partitioning allows the application to be seamlessly integrated into corporate Webs (such as Intranets). In Herrmann, a form is implemented as an application page and is published as an ActiveX object. An application can be

viewed as a collection of discrete frames or views. For example, a typical business application generally includes a set of menu items, each of with invokes a particular frame (a form which manifests certain functionality of the application). In this manner a user can select a web page link from within a browser, so as to invoke a particular frame of the application (i.e., sub-application).

Figures 4A-D show an example of a web-delivered application from the perspective of an end user. Herrmann teaches that from an end user's desktop, the entry point into the system is through a corporate home page that includes a number of links. In response to the user clicking on a particular link to an application page, the web browser connects to the application page (file) residing on the server. Clicking on a link can bring the user to a departmental homepage that includes links pointing to HTML documents that contain policies, status, procedures, reports, and other documents. Some pages can contain HTML forms that are intended to send data to a server-based application

Herrmann also teaches that in addition to the features of an Intranet, a new page type, an application (appdoc) page, is also provided for in-place execution of an application in the Web browser. Since each application page is located using an URL, other pages can have hyperlinks to it. Multiple application pages can be grouped together by making a catalog page that contains hyperlinks to the application pages. When the user selects a hyperlink that points to an application page, the Web browser downloads the application code and executes the page inside the browser. Upon the browser downloading the application page, the browser (based on the defined MIME type) invokes a local handler which is a handler for documents of a type. More particularly, the application page includes a Globally Unique Identifier (GUID) and a codebase URL for identifying a remote (downloadable) application to invoke for hosting the document. Given the document object and the GUID, which arrive with the application page, the local handler looks to the client machine to see if the hosting application already resides locally (e.g., by examining the Windows registry). At this point the local handler can choose to invoke a local copy (if any) or download the latest version of the host application.

When code is downloaded, a "code base" specification (file) is initially requested from the server. The code base itself can range from a simple DLL file to a Cabinet file containing multiple compressed files. Still further, an Information file can be employed for instructing the client system how to install the downloaded application. The machinery employed for actually downloading program code itself relies on standard Microsoft ActiveX API (Application Programming Interface) calls. Although the ActiveX API does not provide native support for Web-delivered applications, its API can be invoked for locating the correct version of the program code, copying it to the local machine, verifying its integrity, and registering it with the client's operating system. Once the code has been downloaded, the handler can proceed to invoke the now-present application host for rendering the document object (in a manner similar to invoking the hosting application through the registry if it were already installed). Once the hosting application (OLE server) is loaded at the client, the client system can employ the OLE document view architecture to render the application correctly within the browser.

New claim 21, on the other hand, recites a method that includes:

> determining that a user has encountered a problem with a webpage currently being viewed by the user with a web browser;
>
> dynamically displaying a widget on the webpage in response to determining that the user has encountered the problem with the webpage;
>
> determining that the user has selected the widget;
>
> activating, in response to determining that the user has selected the widget, a control module embedded in the web page and associated with the widget that is currently being viewed by the web browser, the control module being embedded in the webpage by the web browser when the webpage is loaded into the browser, the activating step comprising:
>
> determining that at least one error associated with form information exists in at least one form field on the web page currently being viewed by the web browser; and
>
> in response to the determining that the at least one error exists, automatically activating the control module without user intervention;
>
> receiving, in response to the activating step, source code from the remote computer, the source code including instructions directing the control module to perform at least one operation on the web browser;
>
> after the activation step and before the receiving step, sending a message from the control module to the remote computer to establish a connection between the control module and a support application executing on the remote computer;

executing, via the control module, the source code received from the remote computer;

reading and modifying, via the control module in response to the executing step, state information of the web browser as instructed by the source code, the state information of the web browser including form information entered into at least one form field of the web page that is currently being viewed by the web browser;

executing the source code so as to read the form information;

sending the form information that is read to the remote computer;

executing the source code so as to modify at least one form field of the web page that is currently being viewed by the web browser;

executing the source code so as to replace information in at least one form field of the web page that is currently being viewed by the web browser; and

executing the source code so as to cause the web browser to view a different web page,

wherein if the state information is read by the control module, the at least one error is recoded, and data associated with the at least one error is sent to the remote computer, and

if the state information is modified by the control module, the form information associated with the at least one error is modified so that the form information is acceptable by the form field.

According to the Examiner, Herrmann teaches activating a control embedded in a web page that is currently being viewed by a web browser, receiving source coded from the remote computer, and using the control to execute the source coded received from the remote computer so as to read and/or modify state information of the web browser. Applicant respectfully traverses this position of the Examiner.

The recited "control module" is a customer support control module that is loaded into a webpage by the web browser as the webpage is loaded. See Specification at page 8, lines 4-11. The control module allows a customer support representative or computerized agent to assist the user with problems being experienced on the webpage. The control module is loaded into the webpage by the web browser while the webpage is being loaded. According to the Examiner, Herrmann teaches "from within a browser, a user would select a Web Page link…invoke a particular frame of the application". That is, a user is required to click a link in order to invoke a sub-application. Therefore, this sub-application is not embedded within the webpage by the web browser when the webpage is loaded.

Furthermore, the source code received by the control module is received, in response to the activation of the control module, and this source code includes instructions directing the control module to perform at least one operation on the web browser. According to the Examiner, Herrmann teaches "download program code". However, a careful reading of the entire paragraph that includes the section of Herrmann cited by the Examiner reveals that Herrmann does not teach or suggest receiving, in response to the activation of the control module, source code from the remote computer, with the source code including instructions directing the control module to perform at least one operation on the web browser. In Herrmann, a new "application page" MIME type includes information that is necessary to create a document (ActiveX document) locally, but also includes information necessary to find the program code for rendering the view of the document. See Herrmann at Col. 8, lines 34-38. This is completely different than, and does not even suggest, receiving, in response to the activation of the control module, source code from the remote computer, with the source code including instructions directing the control module to perform at least one operation on the web browser.

Herrmann further teaches that when a user clicks on a hyperlink to an application page, the web browser downloads the application code and executes the page inside the browser. When the application page is downloaded a local handler is invoked which invokes a local copy of a host application or downloads the host application. The handler can invoke the host application for rendering the document object so that a user can use a rendered application within a web browser. See Herrmann at Col. 9, lines 39-55. However, Herrmann is completely silent on receiving, in response to the activation of the control module, source code from the remote computer, with the source code including instructions directing the control module to perform at least one operation on the web browser.

Herrmann also does not teach or suggest executing, via the control module, the source code received from the remote computer, reading and/or modifying, via the control module, state

information of the web browser as instructed by the source code. As explained above, the portion of Hermann cited by the Examiner merely states the following:

> On the server side, since the .APP file is really a file, the Web server simply receives the request and returns the file to the client. When the APP file is downloaded, the APP DocObject handler asks the operating system to download the codebase for the object specified in the .APP file. This system functionality is available in Windows through the CoGetClassObjectFromURL function. After the ActiveX object's codebase is downloaded, the .APP DocObject handler asks the browser to create a view on itself, for instance, by calling the ActivateMe method on the Explorer document site. The Internet Explorer then calls the DocObject back to instantiate a view, which it does by creating an instance of the ActiveX view object from the code that was downloaded. Once created, the ActiveX view object gets in-place activated in the Internet Explorer, which creates the Delphi form and all its child controls.
>
> Once the form is created, it can establish connections back to any remote server objects it needs to perform its functions.

Nowhere does Herrmann teach that a control module executes source code that directs it to perform one or more operations on the web browser. Furthermore, nowhere does Herrmann teach reading and/or modifying, via the control module, state information of the web browser as instructed by the source code. Herrmann is completely silent on reading or modifying state information of the web browser. The portions of Herrmann cited by the Examiner do not even suggest this claimed feature of the present invention.

Furthermore, nowhere does Herrmann teach or suggest determining that a user has encountered a problem with a webpage currently being viewed by the user with a web browser; dynamically displaying a widget on the webpage in response to determining that the user has encountered the problem with the webpage; determining that the user has selected the widget; activating, in response to determining that the user has selected the widget, a control module embedded in the web page and associated with the widget that is currently being viewed by the web browser, the control module being embedded in the webpage by the web browser when the webpage is loaded into the browser, with the activating step comprising: determining that at least one error associated with form information exists in at least one form field on the web page currently being viewed by the web browser; and in response to the determining that the at least one error exists,

automatically activating the control module without user intervention…executing the source code so as to replace information in at least one form field of the web page that is currently being viewed by the web browser…

Herrmann further does not teach or suggest determining that at least one error associated with form information exists in at least one form field on the web page currently being viewed by the web browser; and in response to the determining that the at least one error exists, automatically activating the control module without user intervention. Herrmann is completely silent as to such a feature. Herrmann is also silent on the feature that if the state information is read by the control module, the at least one error is recoded, and data associated with the at least one error is sent to the remote computer, and if the state information is modified by the control module, the form information associated with the at least one error is modified so that the form information is acceptable by the form field.

Applicant believes that the differences between Herrmann, Lee, and the present invention are clear in new claims 21-25, which recite various embodiments of the present invention. Therefore, claims 21-25 distinguish over the Herrmann and Lee references.

Furthermore, Herrmann does not teach or suggest that the state information of the web browser includes form information entered into at least one form field of the web page that is currently being viewed by the web browser, or executing the source code so as to read the form information; and sending the form information that is read to the remote computer. According to the Examiner, Herrmann teaches that the "user can interact with the form" and "HTML forms that are intended to send data to a server-based application".

A form in Herrmann is implemented as an application page and is published as an ActiveX object. Assuming arguendo that the forms of Herrmann and the forms of the present invention are equivalent, a user interacting with a form has nothing to do with "executing the source code so as to read the form information; and sending the form information that is read to the remote computer".

Furthermore, the HTML form of Herrmann is a typical form that a user can fill out and submit to send the information to a server. This is completely different than a control module "executing the source code so as to read the form information; and sending the form information that is read to the remote computer".

Applicant is not conceding in this application that the canceled claims are not patentable over any prior art, as the present claim amendments and cancelations are only for facilitating expeditious prosecution of allowable subject matter. Applicants respectfully reserve the right to pursue the canceled and other claims in one or more continuation and/or divisional patent applications.

Applicant has examined the references cited by the Examiner as pertinent but not relied upon. It is believed that these references neither disclose nor make obvious the invention recited in the present claims. In view of the foregoing, it is respectfully submitted that the application and the claims are in condition for allowance. Reexamination and reconsideration of the application, as amended, are requested.

If for any reason the Examiner finds the application other than in condition for allowance, the Examiner is invited to call the undersigned attorney at (561) 989-9811 should the Examiner believe a telephone interview would advance the prosecution of the application.

Respectfully submitted,

Date: June 20, 2008

By: /Stephen Bongini/
Stephen Bongini
Reg. No. 40,917
Attorney for Applicant

FLEIT GIBBONS GUTMAN
BONGINI & BIANCO P.L.
One Boca Commerce Center
551 Northwest 77th Street, Suite 111
Boca Raton, Florida 33487
Telephone: (561) 989-9811
Facsimile: (561) 989-9812